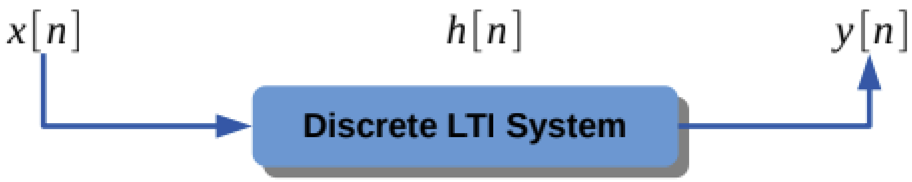
Experiment 5

Title : To study Linear Convolution of finite sequences (finding the LTI system output)

Objective :

1. Algorithm of Linear Convolution between two finite sequences
2. Find the LTI system output while x[n] and h[n] are given as system input and impulse response sequence.
   (a) Plot the impulse response.
   (b) Plot input signal.
   (c) Plot the output signal.

Block Diagram :



$$y[n]=x[n]*h[n]$$

Code 1) :

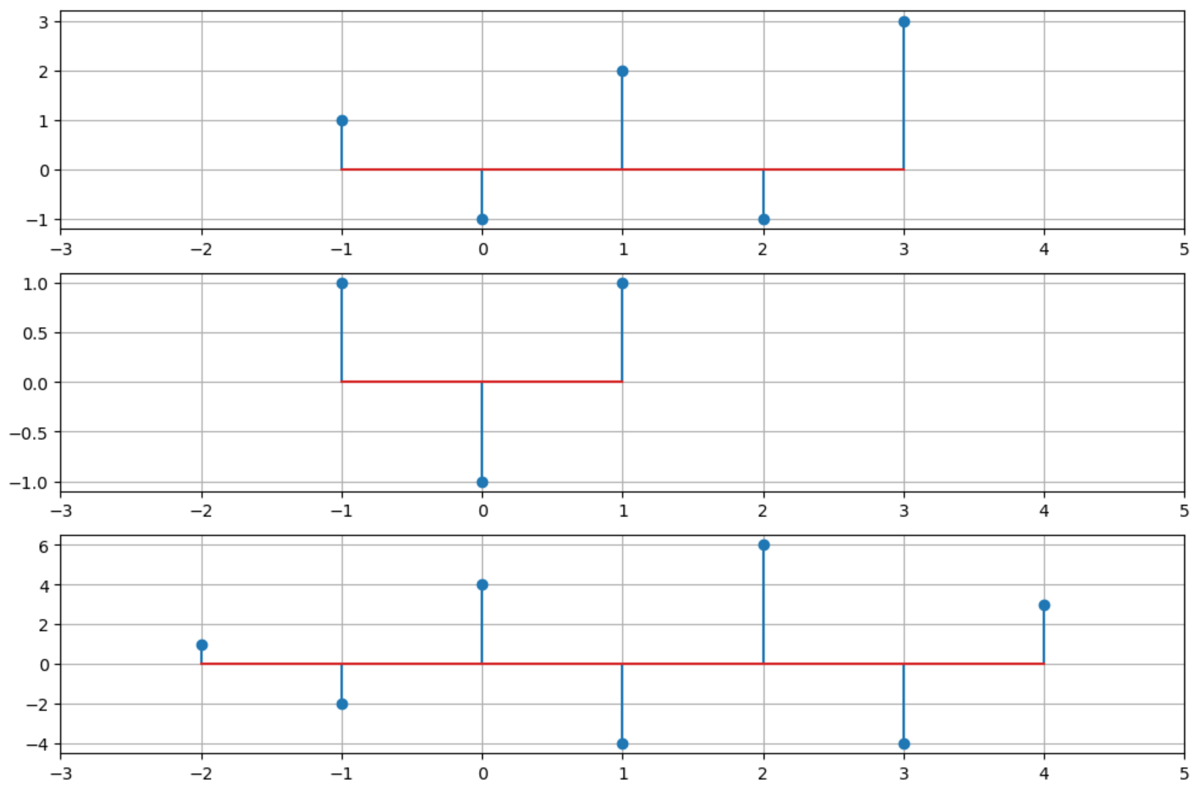
```
import numpy as np
import matplotlib.pyplot as plt

# Define the input sequence xn
xn = np.array([1, -1, 2, -1, 3])

# Get the size (length) of the input sequence
lx = xn.size

# Define the initial index for the input sequence
ix = -1

# Define the impulse response hn
hn = np.array([1, -1, 1])
```

```
# Get the size (length) of the impulse response
lh = hn.size

# Define the initial index for the impulse response
ih = -1

# Calculate the length of the output sequence
ly = lx + lh - 1

# Calculate the initial index for the output sequence
iy = ix + ih

# Initialize an array to store the output sequence
yn = np.zeros(ly)

# Perform linear convolution using nested loops
for i in range(lx):
for j in range(lh):
# Accumulate the convolution result at the appropriate index
yn[i + j] += (xn[i] * hn[j])

# Print the input, impulse response, and output sequences
print("Input = ", xn)
print("Impulse response = ", hn)
print("Output = ", yn)

# Create arrays for the indices of the input, impulse response, and output
sequences
nx = np.arange(ix, ix + lx)
nh = np.arange(ih, ih + lh)
ny = np.arange(iy, iy + ly)

# Create a figure for plotting with specified dimensions
plt.figure(figsize=(12, 8))

# Plot the input sequence as a stem plot in the first subplot
plt.subplot(3, 1, 1)
plt.stem(nx, xn, label="Input")
plt.xlim([min(ny) - 1, max(ny) + 1])
plt.grid()

# Plot the impulse response as a stem plot in the second subplot
plt.subplot(3, 1, 2)
plt.stem(nh, hn, label="Impulse response")
plt.xlim([min(ny) - 1, max(ny) + 1])
plt.grid()
```

```
# Plot the output sequence as a stem plot in the third subplot
plt.subplot(3, 1, 3)
plt.stem(ny, yn, label="Output")
plt.xlim([min(ny) - 1, max(ny) + 1])
plt.grid()

# Display the entire plot
plt.show()
```

Output :

Input = [1 -1 2 -1 3]
Impulse response = [1 -1 1]
Output = [1. -2. 4. -4. 6. -4. 3.]



Code 2):

```
#Day 5
import numpy as np
import matplotlib.pyplot as plt

Ix = 0#4
Ih = 0#-1


#Linear Convolution
#x = np.array([1,0,-1,3,-2,2])
```

```
#h = np.array([1,-1,1])

x = np.random.rand(500)
h = np.ones(21)/21

Lx = x.size #length of input response
Lh = h.size #length of impulse response

print("x[n] sequence length : {}".format(Lx))
print("x[n] sequence length : {}".format(Lh))

Iy = Ix + Ih #starting index of output response
Ly = Lx + Lh - 1 #length of output response

y = np.zeros(Ly) #initializing y array with zeros

for i in range(Lx): #looping for creating output response y
for j in range(Lh):
y[i + j] += x[i]*h[j]

print("input sequence : {}".format(x))
print("IR sequence : {}".format(h))
print("Output sequence : {}".format(y))

nx = np.arange(Ix,Ix + Lx) #bringing all the responses to equal sampling
distance
nh = np.arange(Ih,Ih + Lh)
ny = np.arange(Iy,Iy + Ly)

plt.figure(figsize = (12,8))

plt.subplot(3,1,1)
plt.plot(nx,x)
plt.xlim([min(ny) - 1, max(ny) + 1]) #setting the limit for x-axis
plt.ylim([-5,5])
plt.grid()

plt.subplot(3,1,2)
plt.stem(nh,h)
plt.xlim([min(ny) - 1, max(ny) + 1])
plt.grid()

plt.subplot(3,1,3)
#plt.stem(ny,y)
plt.plot(ny,y)
plt.xlim([min(ny) - 1, max(ny) + 1])
plt.ylim([-5,5])
```

```
plt.grid()

plt.tight_layout()
plt.show()
```

Output :